SparkFun USB-C Host Shield Hookup Guide

none

SparkFun Electronics®

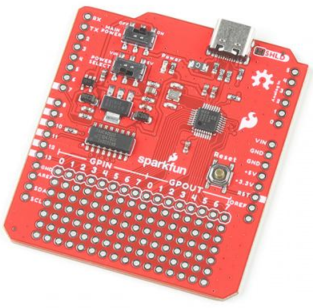
Table of contents

1. Getting Started

1.1 Introduction

⚠ **Attention**

This guide is specific to the USB-C Host Shield board variant. For the variants with the USB (Type-A) connector, please refer to this guide by Hardware Fun.



The SparkFun USB-C Host Shield has similar features to our previous USB Host Shield (v2), but we upgraded the USB Type-A connector to a USB-C connector. Additionally, the board provides users with the option to select either the `5V` or `VIN` pin to power the shield and USB port.

The SparkFun USB Host Shield contains all of the digital logic and analog circuitry necessary to implement a USB peripheral/host controller with your Arduino board. This means you could use your Arduino microcontroller to interface with and control any USB 2.0 compatible device - flash drives, digital cameras, Bluetooth dongles, and much more!

A four-wire serial interface is used to communicate with the host controller chip, so the shield connects the Arduino's hardware SPI pins (D10-13) to the MAX3421E. While the logic-level for the shield is 3.3V, all the SPI signals are sent through a hex converter to keep the shield compatible with any 5V Arduino boards.

Purchase from SparkFun

1.1.1 Required Materials

To get started with the USB-C Host Shield, users will need a few additional items. Users may already have some of these items, feel free to modify your cart accordingly. For users just getting started with electronics, we have linked a few tutorials to establish a foundation of knowledge to follow along with this hookup guide.

- Computer with an operating system (OS) that is compatible with all the software installation requirements.
- A compatible microcontroller/Arduino board; we recommend the SparkFun RedBoard Plus.

> ⚠ **Warning**
>
> The recommended Arduino library for the USB Host Shield is not compatible with all microcontrollers or boards. For a complete list of compatible microcontrollers and boards, please refer to the `README.md` file of USB Host Library Rev. 2.0.

- USB 3.1 Cable A to C - 3 Foot - Used to interface with the RedBoard Plus (1)

a. If your computer doesn't have a USB-A slot or your microcontroller/Arduino board has a different USB connector, then choose an appropriate cable or adapter.

- SparkFun USB-C Host Shield
- USB Peripheral Device *(i.e. flash drive, game controller, smartphone, etc.)* (1)

a. An adapter or cable may be necessary to interface with the peripheral device.

- Headers - Used to connect the shield to the Arduino board (1)

a. Check out some of the options for the Arduino R3/Uno form factor boards below; otherwise, click here for a full selection of our available headers.

- Soldering Tools (1)

a. Check out the beginner tool kit below; otherwise, click here for a full selection of our available soldering tools.

USB 3.1 Cable A to C - 3 Foot

CAB-14743

SparkFun RedBoard Plus

DEV-18158

SparkFun USB-C Host Shield

DEV-21247

Break Away Headers - Straight

PRT-00116

Arduino Stackable Header Kit - R3

PRT-11417

SparkFun Beginner Tool Kit

TOL-14681

Tip

New to soldering? Check out our Through-Hole Soldering Tutorial for a quick introduction!

How to Solder: Through-Hole Soldering

Arduino Examples

The following products are used in the Arduino examples shown in this hookup guide. Users are welcome to choose other products; however, these have been tested and verified to work with the examples.

USB A (Female) to Type C (Male) Converter

COM-21870

USB 2.0 Type-C Cable - 1 Meter

CAB-16905

SparkFun USB Thumb Drive (16GB)

SWG-14658

Bluetooth USB Module Mini

WRL-09434

8BitDo SN30 Pro Bluetooth Gamepad

WIG-17264

Jumper Modification

To modify the jumpers, users will need soldering equipment and/or a knife.

Solder Lead Free - 100-gram Spool

TOL-09325

Weller WLC100 Soldering Station

TOL-14228

Chip Quik No-Clean Flux Pen - 10mL

TOL-14579

Hobby Knife

TOL-09200

Tip

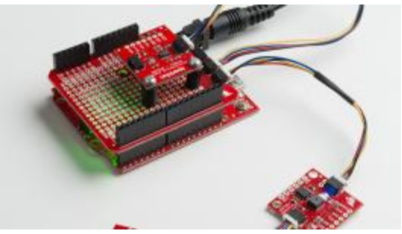
New to jumper pads? Check out our Jumper Pads and PCB Traces Tutorial for a quick introduction!

How to Work with Jumper Pads and PCB Traces

1.1.2 Suggested Reading

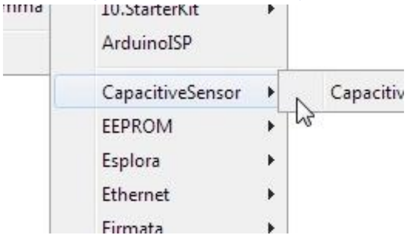
As a more sophisticated product, we will skip over the more fundamental tutorials (i.e. **Ohm's Law** and **What is Electricity?**). However, below are a few tutorials that may help users familiarize themselves with various aspects of the board.

Arduino Shields v2



Installing the Arduino IDE

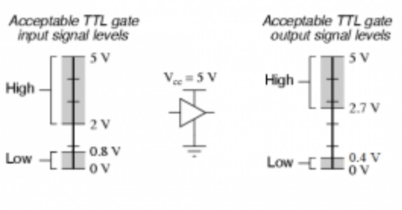
Installing an Arduino Library



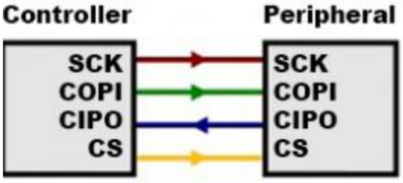
How to Solder: Through-Hole Soldering

How to Work with Jumper Pads and PCB Traces

Logic Levels


Acceptable TTL gate input signal levels
Acceptable TTL gate output signal levels
High
5 V
2 V
Low
0.8 V
0 V
V_{cc} = 5 V
High
5 V
2.7 V
Low
0.4 V
0 V

Serial Peripheral Interface (SPI)



2023-03-04

2023-03-04

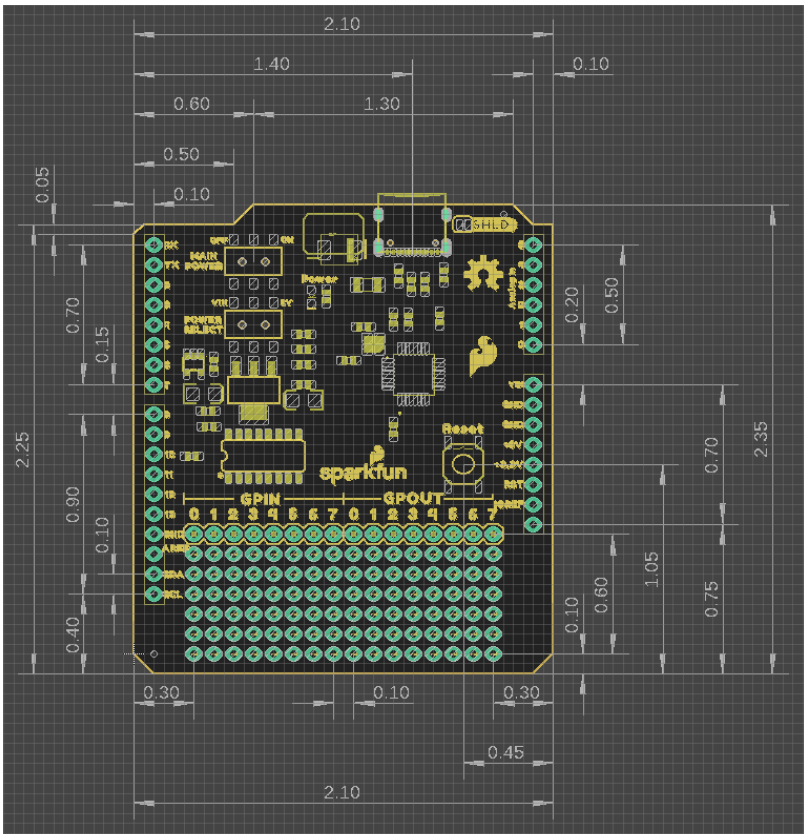
santaimpersonator

GitHub

1.2 Hardware Overview

1.2.1 Board Dimensions

The board dimensions are illustrated in the drawing below; the listed measurements are in inches.



Board dimensions (PDF) for the USB Host Shield, in inches.

Need more measurements?

For more information about the board's dimensions, users can download the eagle files for the board. These files can be opened in Eagle and additional measurements can be made with the dimensions tool.

Download Eagle for Free!

Users can download Eagle for free from AutoDesk.

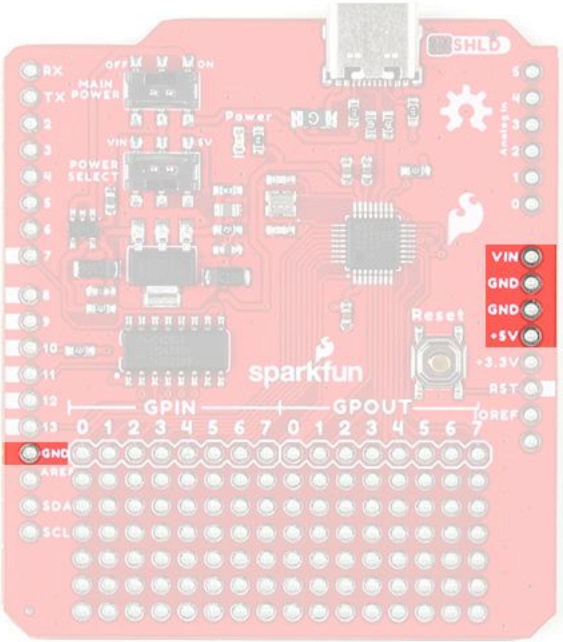
The program is free to use for hobbyists and students. However, it does require an account registration to utilize the software.

Dimensions Tool

This video from Autodesk demonstrates how to utilize the dimensions tool in Eagle, to include additional measurements:

1.2.2 Power

The MAX3421E USB controller only requires **3.3V** to operate; however, the shield *(and USB-C connector)* is powered entirely through either the `5V` or `VIN` pins of the connected Arduino board.



USB Host Shield power connections.

Below, is a general summary of the power circuitry on the board:

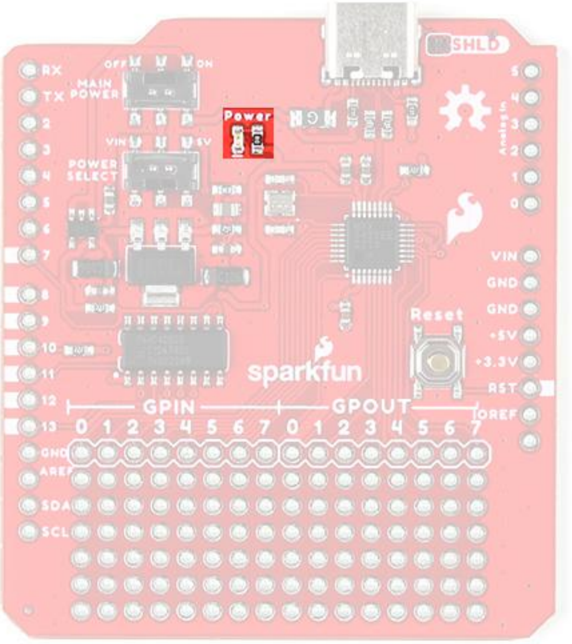
- `VIN` - Provides a regulated 3.3V and 5V for the shield
- To utilize this pin, users will need to connect an external power source to the barrel jack of the Arduino board they are using.
- `5V` - Provides 5V and a regulated 3.3V for the shield
- `GND` - The common ground or the 0V reference for the voltage supplies.
- `VBUS` - *The voltage to the USB-C connector (**5V**)*
- In reference to the `VBUS` net of the schematic.
- The available current is limited to what is supplied from the `VIN` / `5V` pin, up to the 750 mA threshold of the thermal fuse.

> **Info**
>
> When a PD device is connected and the voltage output drops below **4.75V**, the PD device will restrict its current draw to avoid potentially damaging the DFP *(downward-facing port)*.

For more details, users can reference the schematic and the datasheets of the individual components in the power circuitry.

Power LED

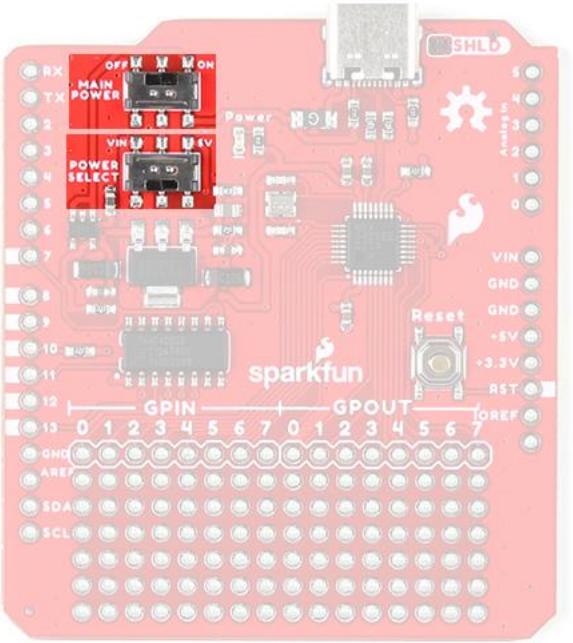
The red, power (`PWR`) LED will light up once **5V** is supplied to the shield. For most users, it will light up when power is supplied to the connected Arduino board.



USB Host Shield `PWR` status LED indicator.

Power Switches

There are two switches on the USB Host Shield. One provides a selectable power input for the shield *(`VIN` or `5V`)* and the other provides power control *(on/off)* to the shield and USB connector.



Power switches on the USB Host Shield.

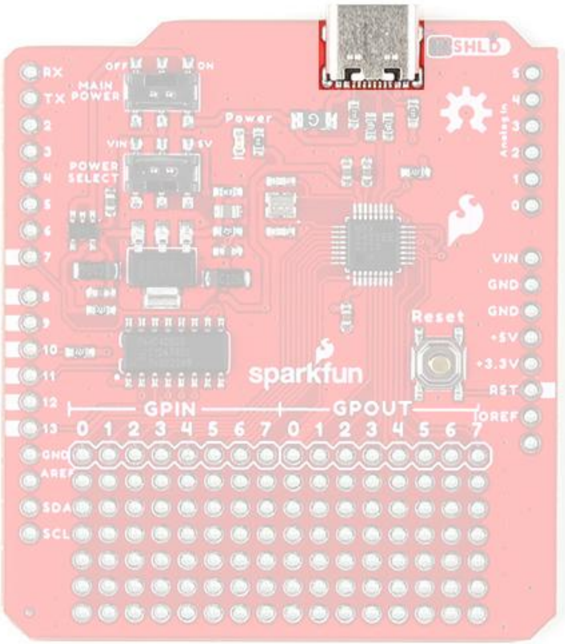
- Power Select
  The power select switch allows users to easily choose the power supply for the shield. This switch mostly controls how the regulated 5V output for the USB-C connector is sourced. However, both options additionally supply the regulated 3.3V for the MAX3421E USB controller.
- **VIN** - Draws power through the Arduino board's `VIN` pin
- Provides a regulated 5V output to the USB-C connector from the `VIN` pin, which is separate/isolated from the `5V` pin of the Arduino board
- Provides a regulated 3.3V output for the MAX3421E USB controller from the regulated 5V output of the `VIN` pin
- **5V** - Draws power through the Arduino board's `5V` pin
- Provides a 5V output to the USB-C connector from the `5V` pin of the Arduino board
- Provides a regulated 3.3V output for the MAX3421E USB controller from the `5V` pin

- Main Power
  The main power switch controls the power input to the shield. This switch turns the shield **on** or **off**; when off, the power output to the USB-C connector is also disabled.

USB-C Connector

> **Charging PD Devices**
>
> When a PD device is connected and the voltage output drops below **4.75V**, the PD device will restrict its current draw to avoid potentially damaging the DFP *(downward-facing port)*.

The USB-C port supports limited power output at **5V**. The available current is limited to what is supplied to the shield from either the `VIN` or `5V` pin, up to the **750 mA** threshold of the thermal fuse.



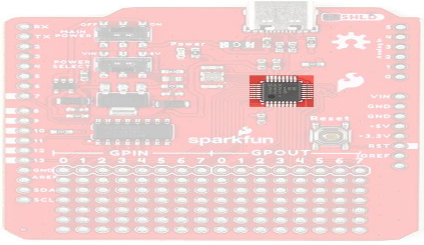
USB-C connector on the USB Host Shield.

1.2.3 USB Controller

The MAX3421E from Maxim Integrated *(now part of Analog Devices)*, is a USB peripheral/host controller that can be implemented as a full-speed USB peripheral or a full-/low-speed host compliant *(USB specification rev 2.0)*. This allows for a vast collection of USB peripherals to be interfaced with an embedded system. The MAX3421E also includes eight general-purpose inputs and outputs so users can reclaim the I/O pins used for the SPI interface and gain additional ones.

Features

- Provides USB Host and Peripheral Functionality
- USB 2.0 Specification: 12 Mbps *(full-speed)*
- 16MB of Embedded SPI Flash Storage
- Operating Voltage: 3.0 - 3.6 V
- Supply Current:
- 45 mA *(max)*
- 8.7 mA *(idle)*
- 30 - 60 µA *(suspend)*
- SPI Clock Speed: 0 - 26 MHz
- Operating Temperature: -40 - +85 °C



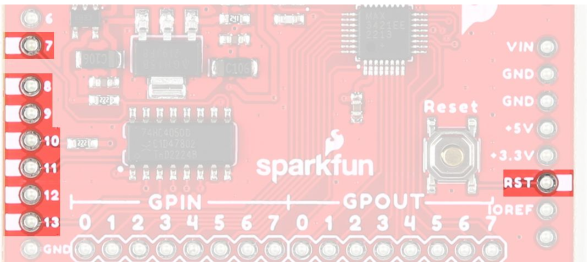
MAX3421E chip on the USB-C Host Shield.

I/O Pins

The MAX3421E is controlled with seven pins on the USB-C Host Shield. Additionally, the MAX3421E provides eight general-purpose inputs and outputs for users to reclaim their I/O pins and gain additional ones.

New Feature

New on this shield, we have added a silkscreen indicator to mark the I/O pins used by the shield. This should help users who are stacking other shields to avoid pin conflicts without referencing the documentation.



I/O pins that are marked on the USB Host Shield.

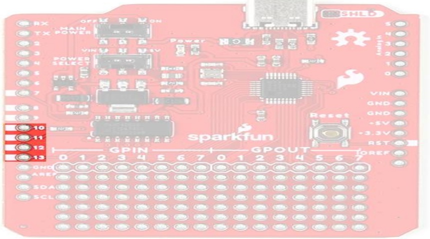
SPI PINS

OSHW Compliance

To comply with the latest OSHW design practices, we have adopted the new SPI signal nomenclature (**SDO/SDI** and **PICO/POCI**). The terms Master and Slave are now referred to as Controller and Peripheral. The `MOSI` signal on a controller has been replaced with `SDO` or `PICO`. Please refer to this announcement on the decision to deprecate the **MOSI/MISO** terminology and transition to the **SDO/SDI** naming convention.

The MAX3421E operates using a register set, accessed by an SPI interface at speeds up to 26MHz. Any SPI controller can add USB peripheral or host functionality using the simple 3- or 4- wire SPI interface The USB-timed operations are performed inside the MAX3421E with interrupts provided at completion, so any SPI controller does not need timers to meet USB timing requirements. Additionally, the firmware to operate the MAX3421E can also be simplified to only support a specific target device.

SCK	`D13` (`SCK`)
SDI or POCI	`D12` (`MISO`)
SDO or PICO	`D11` (`MOSI`)
CS	`D10` (`SS`)

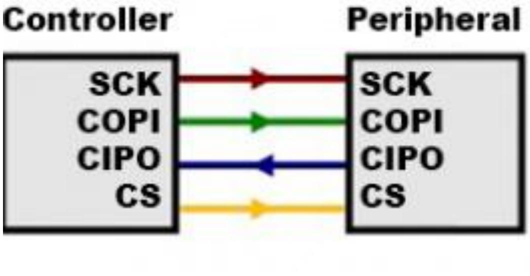


Default SPI bus connections on the USB Host Shield.

Tip

To learn more about the serial peripheral interface (SPI) protocol, check out this great tutorial.

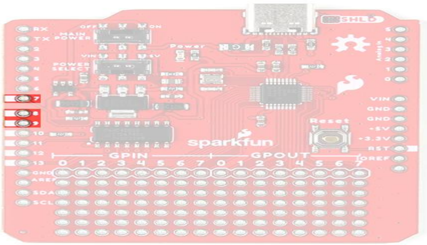
Serial Peripheral Interface (SPI)



I/O PINS

In addition to the SPI pins, there are three I/O pins for the MAX3421E.

INT	`D9` (Output)
GPX	`D8` (Output)
RES	`D7` (Input)



I/O pins on the USB Host Shield.

- `INT` - Interrupt (Output)

  The MAX3421E `INT` pin outputs a signal when a USB event occurs, which requires the attention of the SPI controller. In level mode, the `INT` pin is open-drain and active low. In edge mode, the pin can be operated as push-pull output with programmable polarity. Users can enable the interrupt by setting the IE bit in the CPUCTL (R16) register. The `INT` pin can also be configured to be triggered from the general-purpose inputs (`GPIN0` – `GPIN7`).

- `GPX` - General-Purpose Multiplexed (Output)

  The MAX3421E `GPX` pin indicates one of five internal signals:

- `OPERATE` - The signal is high when the MAX3421E is able to operate after a power-up or `RES` reset.
- `VBUS_DET` - Provides the `VBCOMP` comparator output.
- `BUSACT` - The signal is active (high), whenever there is traffic on the USB bus.
- `INIRQ` - In this mode, `GPIN` interrupts appear only on the `GPX` pin, and do not appear on the `INT` output pin.
- When the SEPIRQ bit of the MODE (R27) register is set high, the `BUSACT` signal is removed
- `SOF` - A square wave is produced, with a positive edge that indicates the USB start-of-frame.

  The internal MAX3421E signal that appears on `GPX` is programmable by writing to the `GPXB` and `GPXA` bits of the PINCTL (R17) register and the `SEPIRQ` bit of the MODE (R27) register.

GPXB	GPXA	GPX PIN OUTPUT
0	0	OPERATE (Default State)
0	1	VBUS_DET
1	0	BUSACT/INIRQ
1	1	SOF

- `RES` - Device Reset (Input)

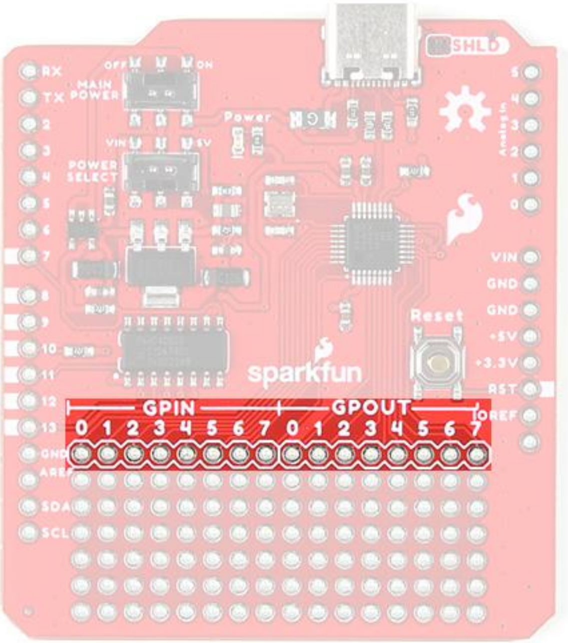
  Driving the `RES` pin low causes a chip reset on the MAX3421E. In a chip reset, all registers are reset to their default states, except for PINCTL (R17), USBCTL (R15), and SPI logic. To bring the MAX3421E out of chip reset, `RES` must be driven high.

> **Note**
>
> The MAX3421E is internally reset if either V_{CC} or V_L is not present. The register file is not accessible under these conditions.

MAX3421E I/O PINS

The MAX3421E also includes eight general-purpose inputs (8) and outputs (8), that can be used to reclaim the I/O pins used for the SPI interface and gain additional ones.

- `GPOUT#` - General-Purpose Push-Pull Outputs.
- `GPIN#` - General-Purpose Inputs.
- `GPIN7` – `GPIN0` are connected to V_L with internal pullup resistors.



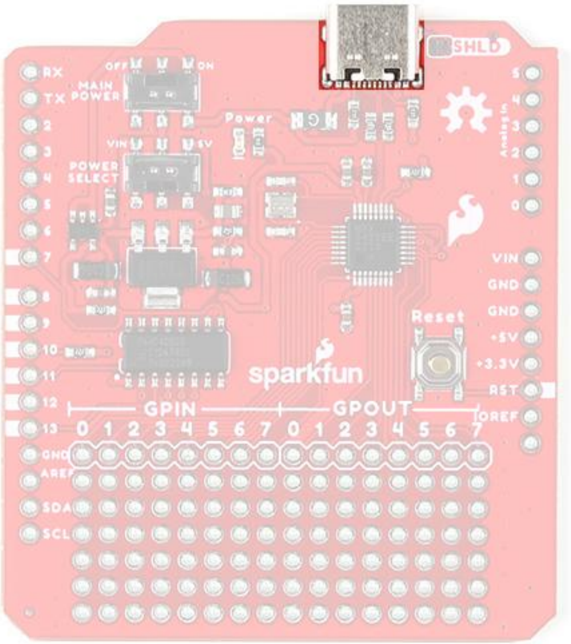
GPIO pins on the USB Host Shield.

USB-C Connector

Charging PD Devices

When a PD device is connected and the voltage output drops below **4.75V**, the PD device will restrict its current draw to avoid potentially damaging the DFP *(downward-facing port)*.

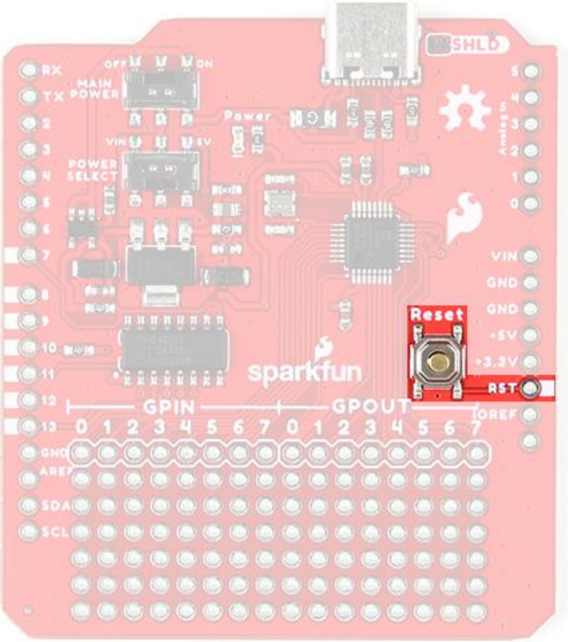
The USB-C connector is used to provide provided an interface to the MAX3421 USB controller, which can function as either a USB peripheral or host. It also supports limited power output at **5V**. The available current is limited to what is supplied to the shield from either the `VIN` or `5V` pin, up to the **750 mA** threshold of the thermal fuse.



USB-C connector on the USB Host Shield.

1.2.4 Reset Button

Sometimes, an Arduino shield covers the `Reset` button of a user's Arduino board; therefore, a `Reset` button is provided on the USB-C Host shield. This allows users to easily reset their Arduino board without having to squeeze in between the Arduino board and shield to hit the button.



`Reset` *button and* `RST` *pin on the USB Host Shield.*

Note

The reset button (`RST` pin) is different from the `RES` (reset) pin for the MAX3421E.

- The button, `RST` pin on the shield, resets the microcontroller of the attached development board.
- The `RES` pin, connected to pin `7` on the shield, is a chip reset for the MAX3421E.

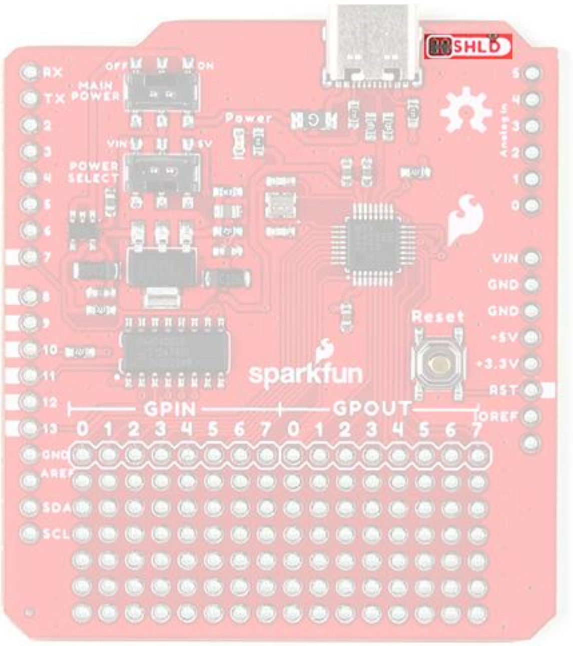
1.2.5 Jumper

There is a **SHLD** jumper on the top of the board that can be used to easily disconnect the shroud of the USB-C connector from `GND`.

Tip

Never modified a jumper before? Check out our Jumper Pads and PCB Traces tutorial for a quick introduction!

How to Work with Jumper Pads and PCB Traces



The `SHLD` jumper on the top of the USB Host Shield.

2023-03-04

2023-03-04

santaimpersonator

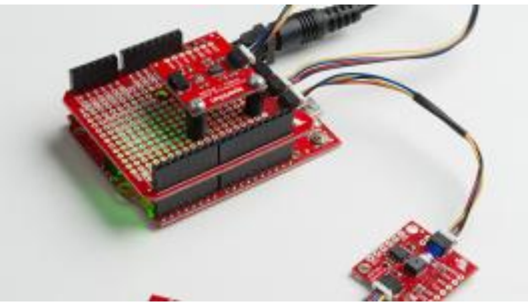
GitHub

1.3 Hardware Assembly

Tip

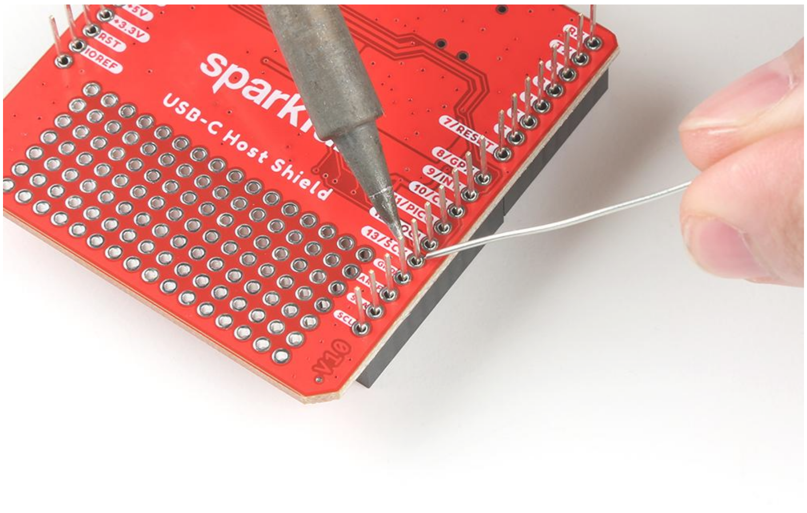
Users unfamiliar with using Arduino shields should refer to the Arduino Shields (v2) tutorial first.

Arduino Shields v2



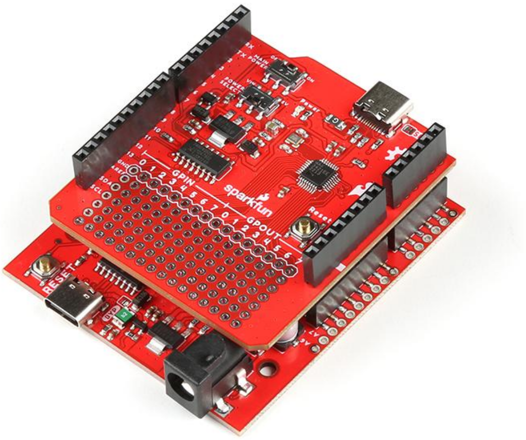
1.3.1 Headers

The pins for the USB Host Shield are broken out to 0.1"-spaced pins on the outer edges of the board. When selecting headers, be sure you are aware of the functionality you need.



Soldering headers to the USB Host Shield.

The Arduino Stackable Header Kit - R3 is a great option as it allows users to stack shields (*w/ Uno/R3 footprint*); with the pins still accessible through the female headers.



Stacking the USB Host Shield on the SparkFun RedBoard Plus.

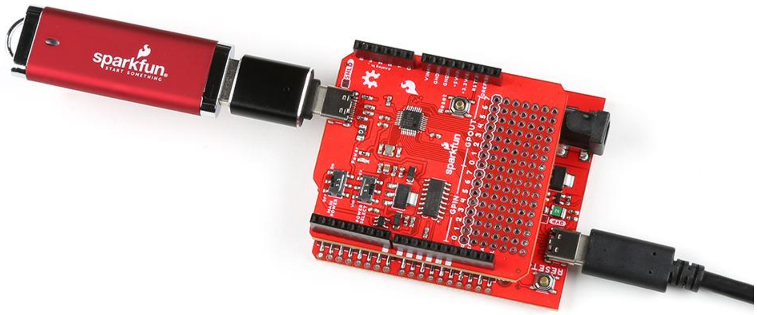
Tip

If you have never soldered before or need a quick refresher, check out our How to Solder: Through-Hole Soldering guide.

How to Solder: Through-Hole Soldering

1.3.2 USB Device

The USB port is utilized for the host/peripheral interface. Users only need to connect a USB device to the USB host shield or connect the shield to a computer with a USB-C cable.



The USB Host Shield with a USB-C adapter and flash drive attached.
The shield sits on top of a RedBoard Plus connected to a computer.

2023-03-04

2023-03-04

santaimpersonator

GitHub

1.4 Software - Arduino IDE

1.4.1 Installation & Setup

Arduino IDE

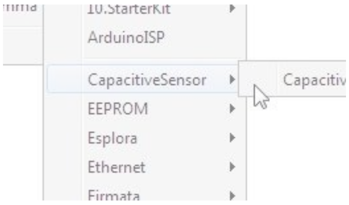
Tip

For first-time users, who have never programmed before and are looking to use the Arduino IDE, we recommend beginning with the SparkFun Inventor's Kit (SIK), which includes a simple board like the Arduino Uno or SparkFun RedBoard and is designed to help users get started programming with the Arduino IDE.

Most users may already be familiar with the Arduino IDE and its use. However, for those of you who have never heard the name *Arduino* before, feel free to check out the Arduino website. To get started with using the Arduino IDE, check out our tutorials below:

WHAT IS AN ARDUINO?

INSTALLING ARDUINO IDE



INSTALLING AN ARDUINO LIBRARY

SparkFun Apollo3 Boards by SparkFun Electronics version [illegible] INSTALLED
Boards included in this package:
SparkFun Artemis Module, SparkFun RedBoard Artemis, SparkFun RedBoard Artemis ATP, SparkFun RedBoard Artemis Nano, SparkFun Artemis Thing Plus, SparkFun Edge, SparkFun Edge2.

SparkFun AVR Boards by SparkFun Electronics version [illegible] INSTALLED
Boards included in this package:
RedBoard, MaKey MaKey, Pro Micro, Fio v3, Qduino Mini, Digital Sandbox, Mega Pro, RedBot, Serial 7-segment Display, ATmega128RFA1 Dev Board, LilyPad USB Plus, SerLCD.

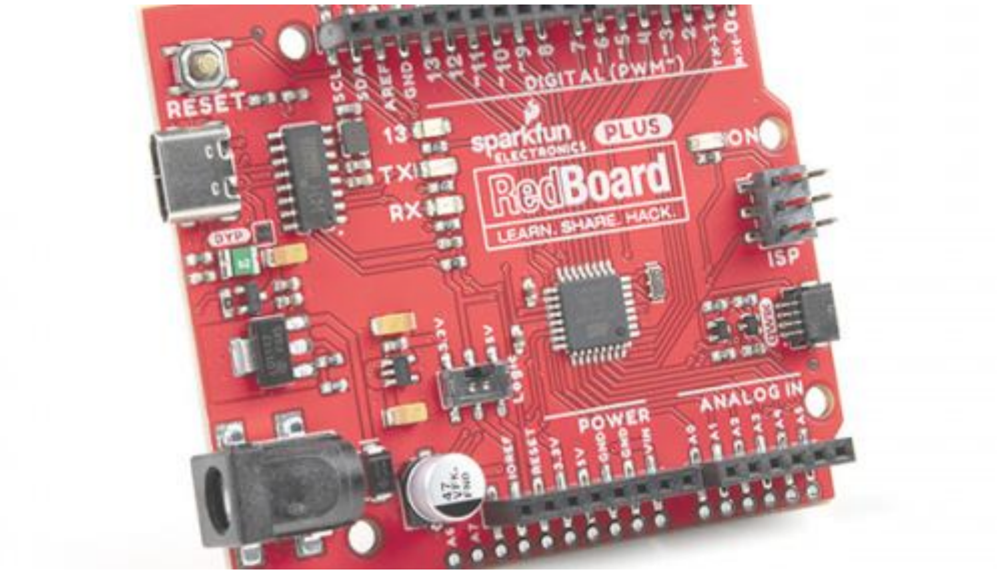
SparkFun SAMD Boards (dependency: Arduino SAMD Boards [illegible]) by SparkFun Electronics version [illegible] INSTALLED
Boards included in this package:
SparkFun SAMD21 Mini Breakout, SparkFun SAMD21 Dev Breakout, SparkFun 9DoF Razor IMU M0, LilyPad LilyMini, SparkFun SAMD21 Pro RF, SparkFun RedBoard Turbo, SparkFun SAMD51 Thing Plus, SparkFun Qwiic Micro.

INSTALLING BOARD DEFINITIONS IN THE ARDUINO IDE

Need help setting up the RedBoard Plus?

REDBOARD PLUS

The following instructions should help users get started with the RedBoard Plus. For more information about the board, please check out our hookup guide below:

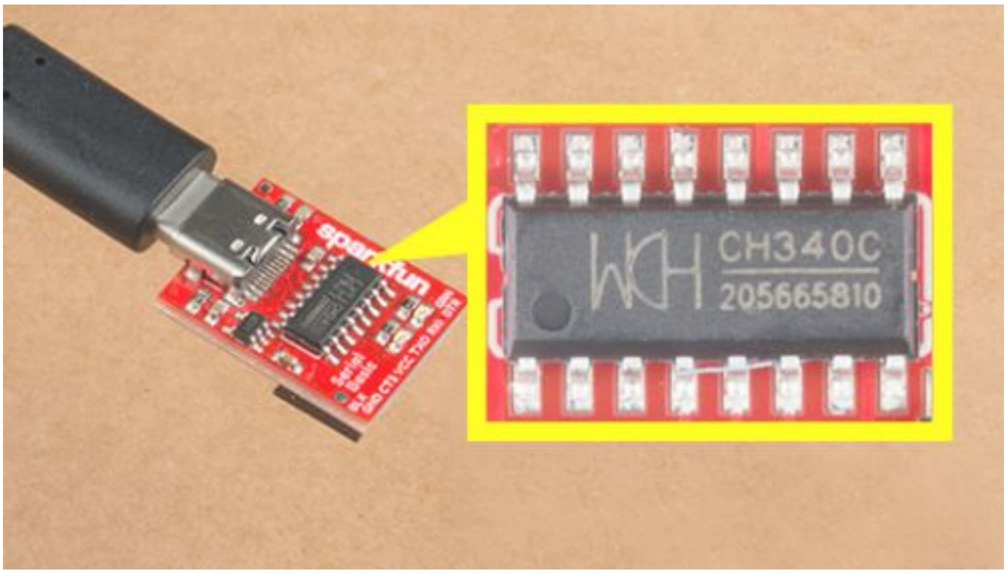


RedBoard Plus Hookup Guide

CH340 Driver

Users will need to install the appropriate driver for their computer to recognize the serial-to-UART chip on their board/adapter. Most of the latest operating systems will recognize the CH340C chip on the board and automatically install the required driver.

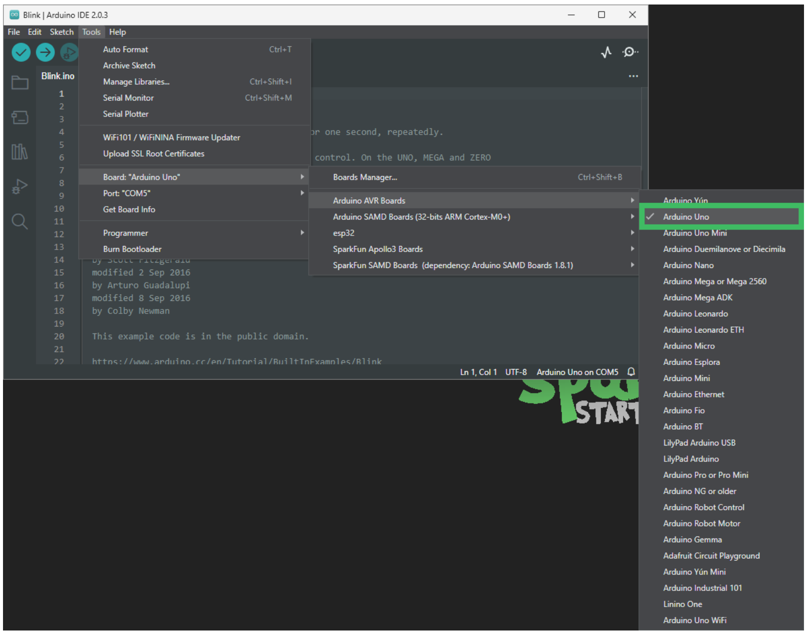
To manually install the CH340 driver on their computer, users can download it from the WCH website. For more information, check out our How to Install CH340 Drivers Tutorial.



How to Install CH340 Drivers

Arduino IDE

When selecting a board to program in the Arduino IDE, users should select the **Arduino Uno** from the **Tools** drop-down menu (_i.e. ***Tools*** > ***Board*** > ***Arduino AVR Boards*** > ***Arduino Uno***).

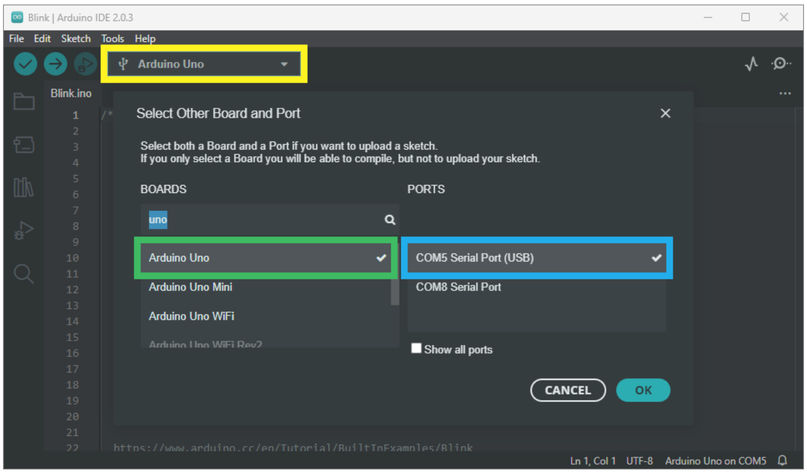


```
or one second, repeatedly.
control. On the UNO, MEGA and ZERO
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
modified 8 Sep 2016
by Colby Newman

This example code is in the public domain.

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
```

*Select the **Arduino Uno** from the Tools drop-down menu in the Arduino IDE.*

Arduino IDE 2.x.x - *Alternative Method*

In the newest version of the Arduino IDE 2.`x`.`x`, users can also select their board (*green*) and port (*blue*) from the `Select Board & Port` dropdown menu (*yellow*).

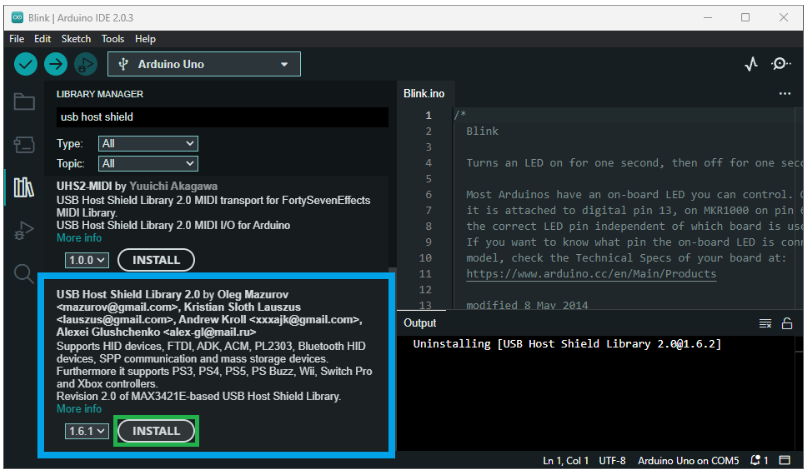


Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

*Selecting the **Arduino Uno** and **COM5** port from the **Select Board & Port** drop-down menu in the Arduino IDE (v2.0.3).*

USB HOST LIBRARY

The USB Host Library Rev. 2.0 can be installed from the library manager in the Arduino IDE.



usb host shield

Type: All

Topic: All

UHS2-MIDI by Yuuichi Akagawa
USB Host Shield Library 2.0 MIDI transport for FortySevenEffects MIDI Library.
USB Host Shield Library 2.0 MIDI I/O for Arduino
More info
1.0.0 INSTALL

USB Host Shield Library 2.0 by **Oleg Mazurov <mazurov@gmail.com>, Kristian Sloth Lauszus <lauszus@gmail.com>, Andrew Kroll <xxxajk@gmail.com>, Alexei Glushchenko <alex-gl@mail.ru>**
Supports HID devices, FTDI, ADK, ACM, PL2303, Bluetooth HID devices, SPP communication and mass storage devices. Furthermore it supports PS3, PS4, PS5, PS Buzz, Wii, Switch Pro and Xbox controllers.
Revision 2.0 of MAX3421E-based USB Host Shield Library.
More info
1.6.1 INSTALL

```
/*
  Blink

  Turns an LED on for one second, then off for one sec

  Most Arduinos have an on-board LED you can control. O
  it is attached to digital pin 13, on MKR1000 on pin 6
  the correct LED pin independent of which board is use
  If you want to know what pin the on-board LED is conn
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
```

```
Uninstalling [USB Host Shield Library 2.0@1.6.2]
```

Ln 1, Col 1 UTF-8 Arduino Uno on COM5 1

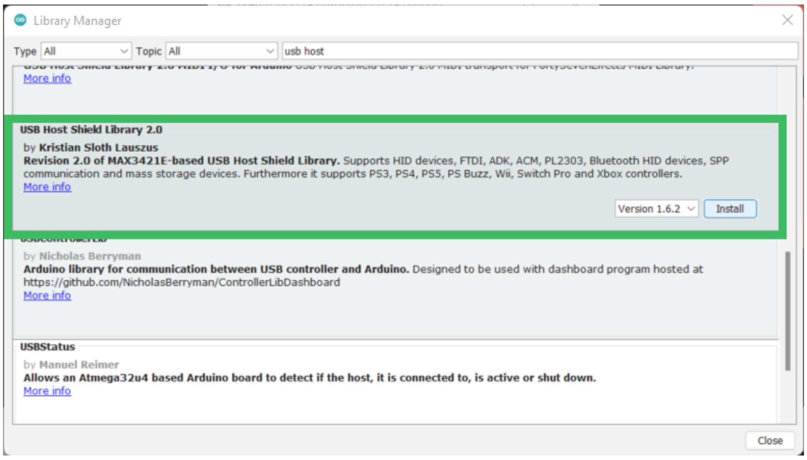
USB Host Library in the library manager of the Arduino IDE.

Info

For more details about the library, check out the online documentation.

Arduino IDE *(v1.x.x)*

In the Arduino IDE v1.x.x, the library manager will have the following appearance for the USB Host Shield library:



Type All Topic All usb host

More info

USB Host Shield Library 2.0
by **Kristian Sloth Lauszus**
Revision 2.0 of MAX3421E-based USB Host Shield Library. Supports HID devices, FTDI, ADK, ACM, PL2303, Bluetooth HID devices, SPP communication and mass storage devices. Furthermore it supports PS3, PS4, PS5, PS Buzz, Wii, Switch Pro and Xbox controllers.
More info
Version 1.6.2 Install

by **Nicholas Berryman**
Arduino library for communication between USB controller and Arduino. Designed to be used with dashboard program hosted at https://github.com/NicholasBerryman/ControllerLibDashboard
More info

USBStatus
by **Manuel Reimer**
Allows an Atmega32u4 based Arduino board to detect if the host, it is connected to, is active or shut down.
More info

Close

USB Host Library in the library manager of the Arduino IDE (v1.x.x).

Alternative Libraries

Users are welcome to try other libraries for the MAX3421E, such as the ones listed below. However, our technical support team will only provide assistance with the USB Host Library Rev. 2.0 recommended in this hookup guide.

- Arduino-Bluetooth
- Lightweight USB Host
- MAX3421E project for STM32

Supported Boards

For a detailed and up-to-date list of boards supported by this library, check out the `README.md` of the GitHub repository:

- All official Arduino AVR boards (Uno, Duemilanove, Mega, Mega 2560, Mega ADK, Leonardo etc.)
- Arduino Due
- Teensy (Teensy++ 1.0, Teensy 2.0, Teensy++ 2.0, Teensy 3.x, Teensy LC and Teensy 4.x)
- For the Teensy 3.x, install this SPI library and add `#include <spi4teensy3.h>` to the `*.ino` sketch file.
- STM32F4
- Take a look at the following example code.
- ESP8266 is supported using the ESP8266 Arduino core
- Uses pins `15` and `5` for `CS` and `INT`, respectively.
- `GPIO6` - `GPIO11` and `GPIO16` are **NOT** usable.
- ESP32 is supported using the arduino-esp32
- `GPIO5` : `CS`
- `GPIO17` : `INT`
- `GPIO18` : `SCK`
- `GPIO19` : `POCI`
- `GPIO23` : `PICO`

I/O Pin Modifications

The SPI pins used by this library are dictated by SPI library for the Arduino core being utilized and cannot be changed easily. It is recommended that the default pins of the SPI library be utilized.

However, the USB Host Library also declares its `CS` and `INT` pins. These pins can be reconfigured in the library by modifying the UsbCore.h file:

```
typedef MAX3421e< "CS Pin", "INT Pin" > MAX3421E;
```

For instance, if a user wanted to reconfigure the `CS` pin to `D7` and the `INT` pin to `D2` of the RedBoard Plus *(or any other Arduino Uno/ATmega328P based board)*, line 58 should read:

```
typedef MAX3421e<P7, P2> MAX3421E;
```

Tip

The information above is an example of a pin modification. However, it is not required for the general use of the shield and the examples in this guide. For more information, please refer to the instructions in the `README.md` of the GitHub repository.

Other Boards

For other boards, users will need to modify the lines based on the microcontroller type. For example, with the SparkFun IoT RedBoard users would need to modify line 52.

2023-03-04

2023-03-04

santaimpersonator

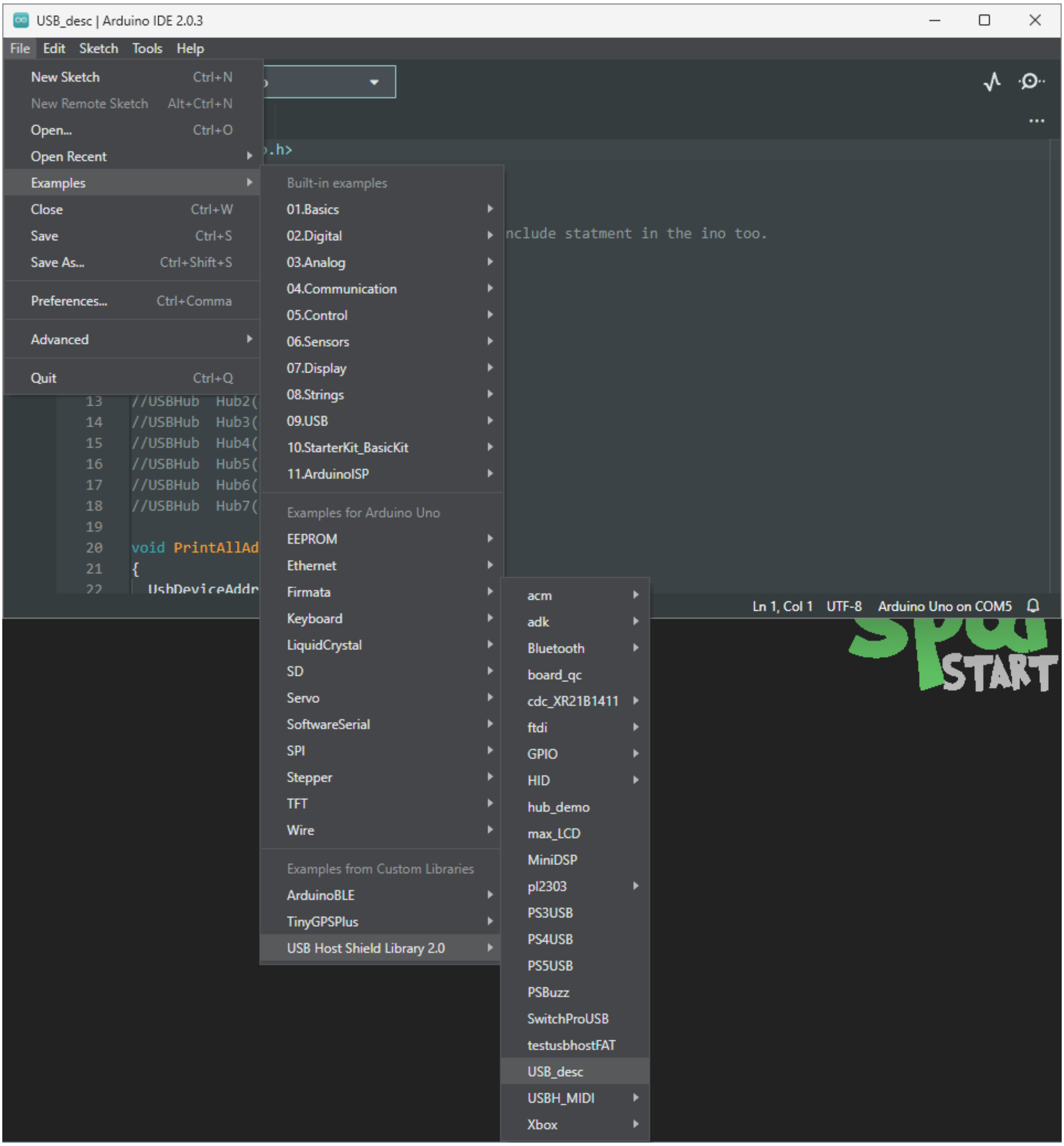
GitHub

1.4.2 Examples

Device Description

USB DESCRIPTION

For our first example, we will be utilizing the USB_dec example from the USB_Host_Shield_2.0 Arduino library. This example can be found in the **File** dropdown menu *(i.e. (1)* ***File > Examples > USB Host Shield Library 2.0 > USB_Desc****)*. Once the example has been opened, users should see two files `USB_desc.ino` and `pgmstrings.h`.

1. 

```
.h>

nclude statment in the ino too.

//USBHub  Hub2(
//USBHub  Hub3(
//USBHub  Hub4(
//USBHub  Hub5(
//USBHub  Hub6(
//USBHub  Hub7(

void PrintAllAd
{
  UsbDeviceAddr
```

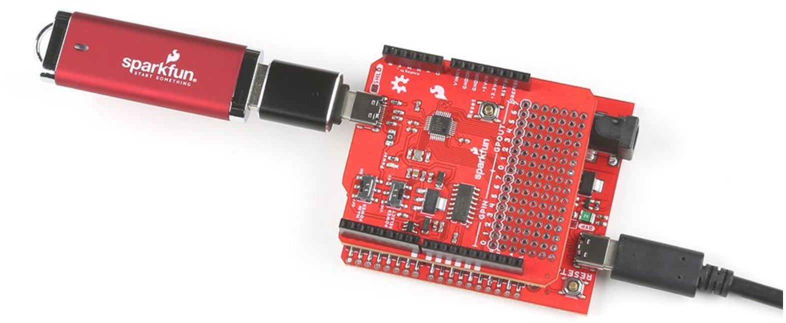
Select the `USB_Desc` example sketch from the `File` drop-down menu.

ample Files

USB_desc.ino pgmstrings.h

```
#include <usbhub.h>

#include "pgmstrings.h"

// Satisfy the IDE, which needs to see the include statment in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB     Usb;
//USBHub  Hub1(&Usb);
//USBHub  Hub2(&Usb);
//USBHub  Hub3(&Usb);
//USBHub  Hub4(&Usb);
//USBHub  Hub5(&Usb);
//USBHub  Hub6(&Usb);
//USBHub  Hub7(&Usb);

void PrintAllAddresses(UsbDevice *pdev)
{
  UsbDeviceAddress adr;
  adr.devAddress = pdev->address.devAddress;
  Serial.print("\r\nAddr:");
  Serial.print(adr.devAddress, HEX);
  Serial.print("(");
  Serial.print(adr.bmHub, HEX);
  Serial.print(".");
  Serial.print(adr.bmParent, HEX);
  Serial.print(".");
  Serial.print(adr.bmAddress, HEX);
  Serial.println(")");
}

void PrintAddress(uint8_t addr)
{
  UsbDeviceAddress adr;
  adr.devAddress = addr;
  Serial.print("\r\nADDR:\t");
  Serial.println(adr.devAddress, HEX);
  Serial.print("DEV:\t");
  Serial.println(adr.bmAddress, HEX);
  Serial.print("PRNT:\t");
  Serial.println(adr.bmParent, HEX);
  Serial.print("HUB:\t");
  Serial.println(adr.bmHub, HEX);
}

void setup()
{
  Serial.begin( 115200 );
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection
#endif
  Serial.println("Start");

  if (Usb.Init() == -1)
    Serial.println("OSC did not start.");

  delay( 200 );
}

uint8_t getdevdescr( uint8_t addr, uint8_t &num_conf );

void PrintDescriptors(uint8_t addr)
{
  uint8_t rcode = 0;
  uint8_t num_conf = 0;

  rcode = getdevdescr( (uint8_t)addr, num_conf );
  if( rcode )
  {
    printProgStr(Gen_Error_str);
    print_hex( rcode, 8 );
  }
  Serial.print("\r\n");

  for (int i = 0; i < num_conf; i++)
  {
    rcode = getconfdescr( addr, i );                 // get configuration descriptor
    if( rcode )
    {
      printProgStr(Gen_Error_str);
      print_hex(rcode, 8);
    }
    Serial.println("\r\n");
  }
}

void PrintAllDescriptors(UsbDevice *pdev)
{
  Serial.println("\r\n");
  print_hex(pdev->address.devAddress, 8);
  Serial.println("\r\n--");
  PrintDescriptors( pdev->address.devAddress );
}

void loop()
{
  Usb.Task();

  if( Usb.getUsbTaskState() == USB_STATE_RUNNING )
  {
    Usb.ForEachUsbDevice(&PrintAllDescriptors);
    Usb.ForEachUsbDevice(&PrintAllAddresses);

    while( 1 ) { // stop
#ifdef ESP8266
        yield(); // needed in order to reset the watchdog timer on the ESP8266
```

Users will need to connect a peripheral USB device to the USB-C connector, before running the example. After the example begins, users should see an output in the Serial Monitor with a description of the connected USB device.



The USB Host Shield with a USB-C adapter and flash drive attached.

USB Hubs

If users connect USB hubs or USB cables with a hub to the USB host shield, utilize the hub_demo example from the USB_Host_Shield_2.0 Arduino library instead. This example can be found in the **File** dropdown menu *(i.e.* ***File > Examples > USB Host Shield Library 2.0 > hub_demo****)* and will list the USB description for the hub(s) and all the peripheral devices connected to the hub(s).

Only interested in the USB hub description?

To see just the USB description for the hub(s) connected to the USB host shield, follow the information in the library's FAQ. Utilizing the USB_dec example, uncomment lines 12-18(1).

1. Each instance of `USBHub Hub<number>(&Usb);` enables a USB hub, but the library is limited up to **seven** USB hubs.

```
11  USB     Usb;
12  //USBHub  Hub1(&Usb);
13  //USBHub  Hub2(&Usb);
14  //USBHub  Hub3(&Usb);
15  //USBHub  Hub4(&Usb);
16  //USBHub  Hub5(&Usb);
17  //USBHub  Hub6(&Usb);
18  //USBHub  Hub7(&Usb);
```

Qwiic USB Hub - USB2514B

SPX-18014

SparkFun 4-in-1 Multi-USB Cable - USB-C Host

CAB-21271

SparkFun 4-in-1 Multi-USB Cable - USB-A Host

CAB-21272

2023-03-04

2023-03-04

santaimpersonator

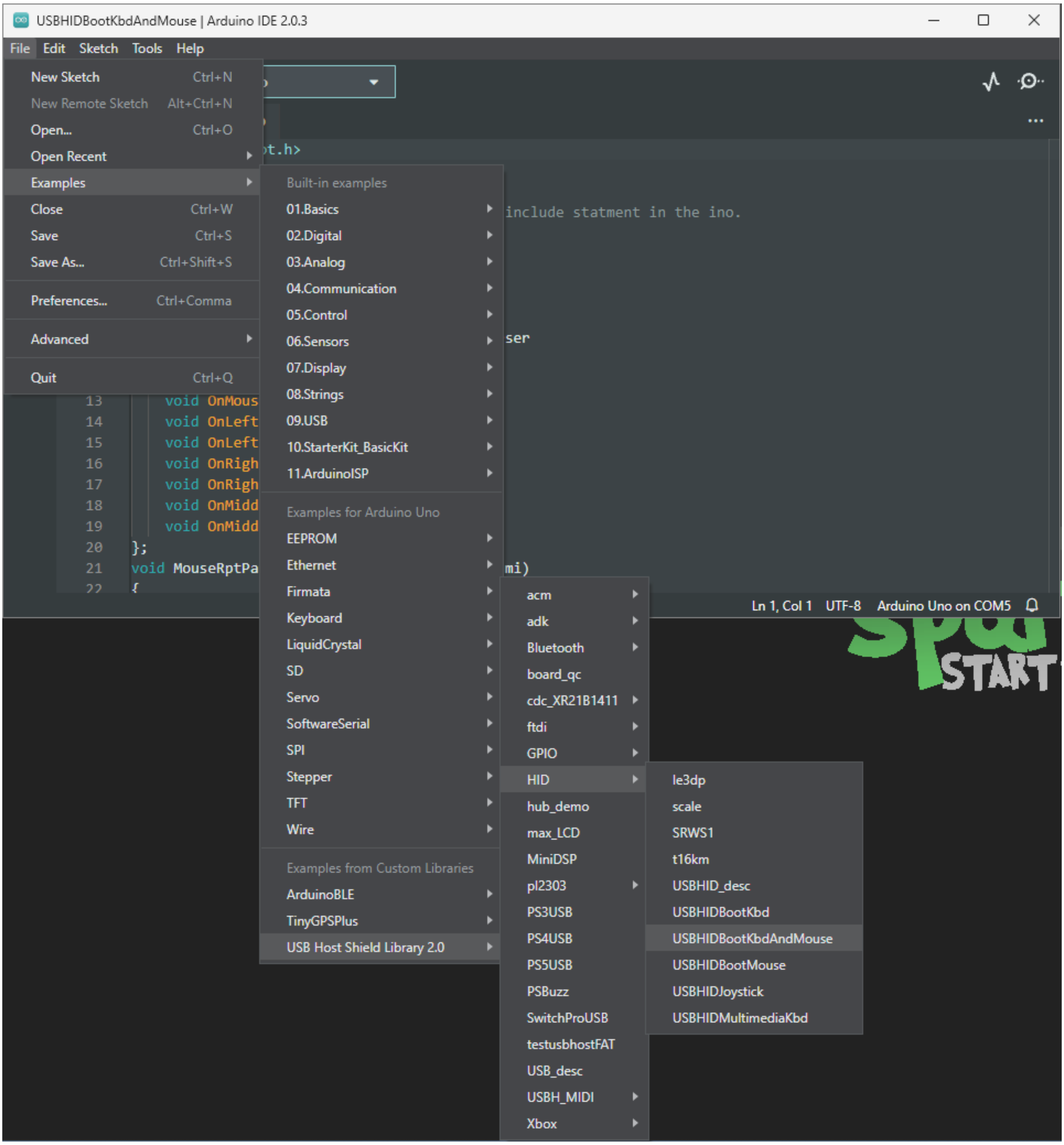
GitHub

Keyboard & Mouse

HID KEYBOARD AND MOUSE

In this example, we will be utilizing the USBHIDBootKbdAndMouse example from the USB_Host_Shield_2.0 Arduino library. This example can be found in the **File** dropdown menu *(i.e. (1)* ***File > Examples > USB Host Shield Library 2.0 > HID >***

USBHIDBootKbdAndMouse). Once the example has been opened, users should see the `USBHIDBootKbdAndMouse.ino` example sketch.

1. 

Select the `USBHIDBootKbdAndMouse` example sketch from the `File` drop-down menu.

BHIDBootKbdAndMouse.ino

```
#include <hidboot.h>
#include <usbhub.h>

// Satisfy IDE, which only needs to see the include statment in the ino.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

class MouseRptParser : public MouseReportParser
{
  protected:
    void OnMouseMove(MOUSEINFO *mi);
    void OnLeftButtonUp(MOUSEINFO *mi);
    void OnLeftButtonDown(MOUSEINFO *mi);
    void OnRightButtonUp(MOUSEINFO *mi);
    void OnRightButtonDown(MOUSEINFO *mi);
    void OnMiddleButtonUp(MOUSEINFO *mi);
    void OnMiddleButtonDown(MOUSEINFO *mi);
};
void MouseRptParser::OnMouseMove(MOUSEINFO *mi)
{
  Serial.print("dx=");
  Serial.print(mi->dX, DEC);
  Serial.print(" dy=");
  Serial.println(mi->dY, DEC);
};
void MouseRptParser::OnLeftButtonUp (MOUSEINFO *mi)
{
  Serial.println("L Butt Up");
};
void MouseRptParser::OnLeftButtonDown   (MOUSEINFO *mi)
{
  Serial.println("L Butt Dn");
};
void MouseRptParser::OnRightButtonUp    (MOUSEINFO *mi)
{
  Serial.println("R Butt Up");
};
void MouseRptParser::OnRightButtonDown  (MOUSEINFO *mi)
{
  Serial.println("R Butt Dn");
};
void MouseRptParser::OnMiddleButtonUp   (MOUSEINFO *mi)
{
  Serial.println("M Butt Up");
};
void MouseRptParser::OnMiddleButtonDown (MOUSEINFO *mi)
{
  Serial.println("M Butt Dn");
};

class KbdRptParser : public KeyboardReportParser
{
    void PrintKey(uint8_t mod, uint8_t key);

  protected:
    void OnControlKeysChanged(uint8_t before, uint8_t after);
    void OnKeyDown  (uint8_t mod, uint8_t key);
    void OnKeyUp    (uint8_t mod, uint8_t key);
    void OnKeyPressed(uint8_t key);
};

void KbdRptParser::PrintKey(uint8_t m, uint8_t key)
{
  MODIFIERKEYS mod;
  *((uint8_t*)&mod) = m;
  Serial.print((mod.bmLeftCtrl   == 1) ? "C" : " ");
  Serial.print((mod.bmLeftShift  == 1) ? "S" : " ");
  Serial.print((mod.bmLeftAlt    == 1) ? "A" : " ");
  Serial.print((mod.bmLeftGUI    == 1) ? "G" : " ");

  Serial.print(" >");
  PrintHex<uint8_t>(key, 0x80);
  Serial.print("< ");

  Serial.print((mod.bmRightCtrl   == 1) ? "C" : " ");
  Serial.print((mod.bmRightShift  == 1) ? "S" : " ");
  Serial.print((mod.bmRightAlt    == 1) ? "A" : " ");
  Serial.println((mod.bmRightGUI    == 1) ? "G" : " ");
};

void KbdRptParser::OnKeyDown(uint8_t mod, uint8_t key)
{
  Serial.print("DN ");
  PrintKey(mod, key);
  uint8_t c = OemToAscii(mod, key);

  if (c)
    OnKeyPressed(c);
}

void KbdRptParser::OnControlKeysChanged(uint8_t before, uint8_t after) {

  MODIFIERKEYS beforeMod;
  *((uint8_t*)&beforeMod) = before;

  MODIFIERKEYS afterMod;
  *((uint8_t*)&afterMod) = after;
  if (beforeMod.bmLeftCtrl != afterMod.bmLeftCtrl) {
    Serial.println("LeftCtrl changed");
  }
  if (beforeMod.bmLeftShift != afterMod.bmLeftShift) {
    Serial.println("LeftShift changed");
  }
  if (beforeMod.bmLeftAlt != afterMod.bmLeftAlt) {
    Serial.println("LeftAlt changed");
  }
  if (beforeMod.bmLeftGUI != afterMod.bmLeftGUI) {
    Serial.println("LeftGUI changed");
  }

  if (beforeMod.bmRightCtrl != afterMod.bmRightCtrl) {
    Serial.println("RightCtrl changed");
  }
  if (beforeMod.bmRightShift != afterMod.bmRightShift) {
    Serial.println("RightShift changed");
  }
  if (beforeMod.bmRightAlt != afterMod.bmRightAlt) {
    Serial.println("RightAlt changed");
  }
  if (beforeMod.bmRightGUI != afterMod.bmRightGUI) {
    Serial.println("RightGUI changed");
  }

}

void KbdRptParser::OnKeyUp(uint8_t mod, uint8_t key)
{
  Serial.print("UP ");
  PrintKey(mod, key);
}

void KbdRptParser::OnKeyPressed(uint8_t key)
{
  Serial.print("ASCII: ");
  Serial.println((char)key);
};

USB     Usb;
USBHub     Hub(&Usb);

HIDBoot < USB_HID_PROTOCOL_KEYBOARD | USB_HID_PROTOCOL_MOUSE > HidComposite(&Usb);
HIDBoot<USB_HID_PROTOCOL_KEYBOARD>    HidKeyboard(&Usb);
HIDBoot<USB_HID_PROTOCOL_MOUSE>    HidMouse(&Usb);

KbdRptParser KbdPrs;
MouseRptParser MousePrs;

void setup()
{
  Serial.begin( 115200 );
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection
#endif
  Serial.println("Start");

  if (Usb.Init() == -1)
    Serial.println("OSC did not start.");

  delay( 200 );

  HidComposite.SetReportParser(0, &KbdPrs);
  HidComposite.SetReportParser(1, &MousePrs);
  HidKeyboard.SetReportParser(0, &KbdPrs);
  HidMouse.SetReportParser(0, &MousePrs);
}

void loop()
{
  Usb.Task();
}
```

Users will need to connect an HID device *(keyboard and/or mouse)* to the USB-C host shield with a USB cable, before running the example. After the example begins, users should see an output in the Serial Monitor with print out based on the user's interaction with their HID device.

2023-03-04

2023-03-04

santaimpersonator

GitHub

Game Controller

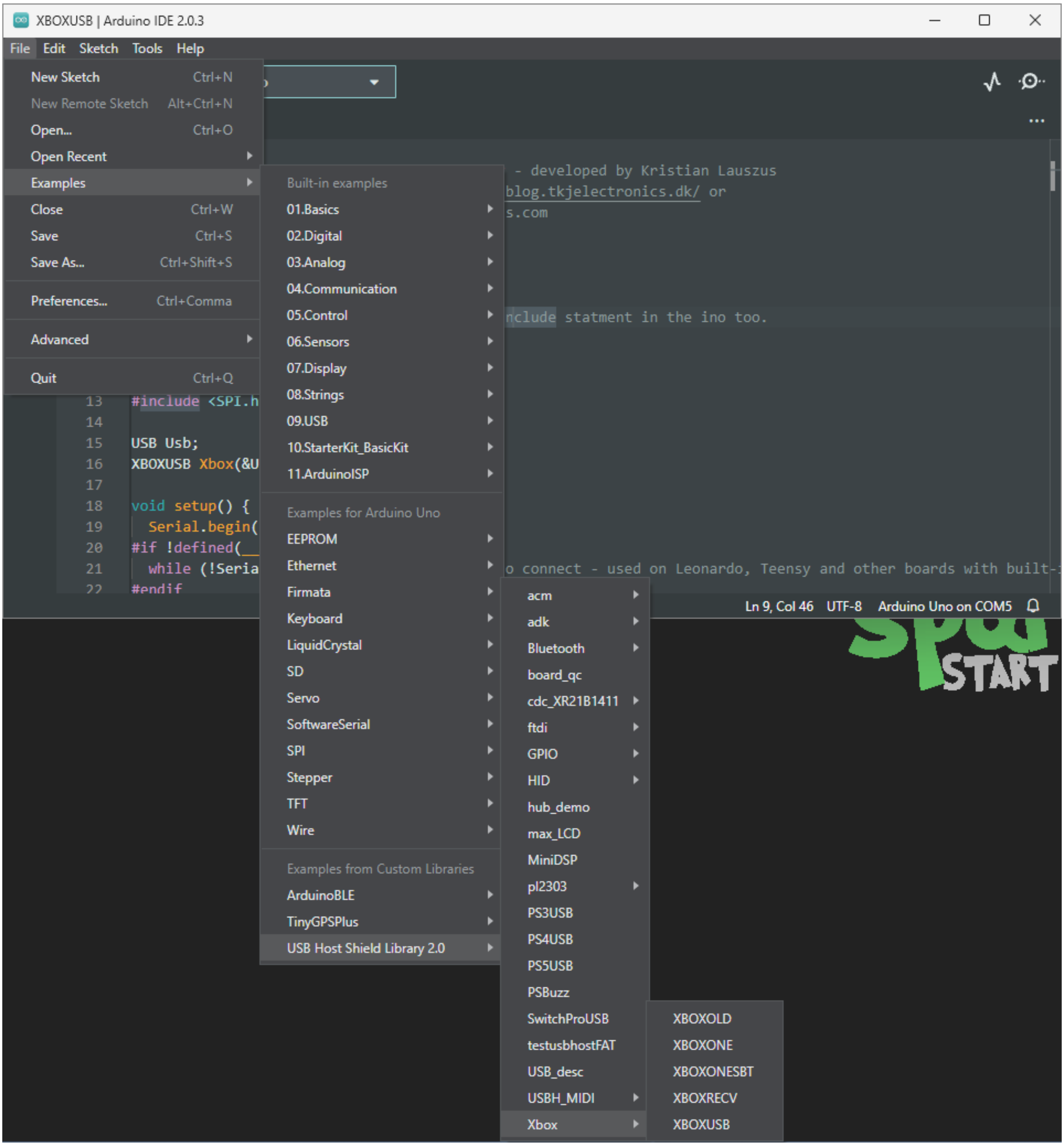
HID GAME CONTROLLER

In these examples, we will be connecting the 8BitDo SN30 Pro to the USB-C host shield. Users will need the following items for the examples below:

- 8BitDo SN30 Pro Bluetooth Gamepad
  For instructions on how to use the 8BitDo SN30 Pro, please refer to their user manual.
- USB 2.0 Type-C Cable - 1 Meter
- USB A (Female) to Type C (Male) Converter
- Bluetooth USB Module Mini

USB Connection

In this example, we will be utilizing the XBOXUSB example from the USB_Host_Shield_2.0 Arduino library. This example can be found in the **File** dropdown menu *(i.e. (1)* ***File > Examples > USB Host Shield Library 2.0 > Xbox > XBOXUSB****)*. Once the example has been opened, users should see the `XBOXUSB.ino` example sketch.

1. 

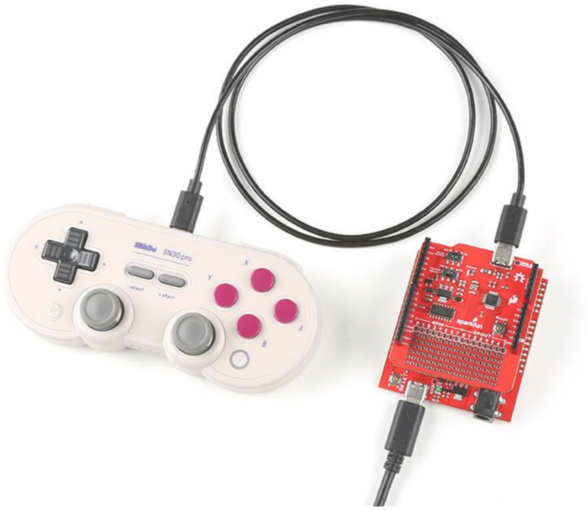
```
 - developed by Kristian Lauszus
blog.tkjelectronics.dk/ or
s.com

nclude statment in the ino too.

#include <SPI.h

USB Usb;
XBOXUSB Xbox(&U

void setup() {
  Serial.begin(
#if !defined(__
  while (!Seria                o connect - used on Leonardo, Teensy and other boards with built-
#endif
```

Select the `XBOXUSB` *example sketch from the* `File` *drop-down menu.*

OXONE.ino

```
/*
 Example sketch for the Xbox 360 USB library - developed by Kristian Lauszus
 For more information visit my blog: http://blog.tkjelectronics.dk/ or
 send me an e-mail:  kristianl@tkjelectronics.com
 */

#include <XBOXUSB.h>

// Satisfy the IDE, which needs to see the include statment in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB Usb;
XBOXUSB Xbox(&Usb);

void setup() {
  Serial.begin(115200);
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection
#endif
  if (Usb.Init() == -1) {
    Serial.print(F("\r\nOSC did not start"));
    while (1); //halt
  }
  Serial.print(F("\r\nXBOX USB Library Started"));
}
void loop() {
  Usb.Task();
  if (Xbox.Xbox360Connected) {
    if (Xbox.getButtonPress(LT) || Xbox.getButtonPress(RT)) {
      Serial.print("LT: ");
      Serial.print(Xbox.getButtonPress(LT));
      Serial.print("\tRT: ");
      Serial.println(Xbox.getButtonPress(RT));
      Xbox.setRumbleOn(Xbox.getButtonPress(LT), Xbox.getButtonPress(RT));
    } else
      Xbox.setRumbleOn(0, 0);

    if (Xbox.getAnalogHat(LeftHatX) > 7500 || Xbox.getAnalogHat(LeftHatX) < -7500 || Xbox.getAnalogHat(LeftHatY) > 7500 || Xbox.getAnalogHat(LeftHatY) < -7500 || Xbox.getAnalogHat(RightHatX) > 7500 || Xbox.getAnalogHat(RightHatX) < -7500 || Xbox.getAnalogHat(RightHatY) > 7500 || Xbox.getAnalogHat(RightHatY) < -7500) {
      if (Xbox.getAnalogHat(LeftHatX) > 7500 || Xbox.getAnalogHat(LeftHatX) < -7500) {
        Serial.print(F("LeftHatX: "));
        Serial.print(Xbox.getAnalogHat(LeftHatX));
        Serial.print("\t");
      }
      if (Xbox.getAnalogHat(LeftHatY) > 7500 || Xbox.getAnalogHat(LeftHatY) < -7500) {
        Serial.print(F("LeftHatY: "));
        Serial.print(Xbox.getAnalogHat(LeftHatY));
        Serial.print("\t");
      }
      if (Xbox.getAnalogHat(RightHatX) > 7500 || Xbox.getAnalogHat(RightHatX) < -7500) {
        Serial.print(F("RightHatX: "));
        Serial.print(Xbox.getAnalogHat(RightHatX));
        Serial.print("\t");
      }
      if (Xbox.getAnalogHat(RightHatY) > 7500 || Xbox.getAnalogHat(RightHatY) < -7500) {
        Serial.print(F("RightHatY: "));
        Serial.print(Xbox.getAnalogHat(RightHatY));
      }
      Serial.println();
    }

    if (Xbox.getButtonClick(UP)) {
      Xbox.setLedOn(LED1);
      Serial.println(F("Up"));
    }
    if (Xbox.getButtonClick(DOWN)) {
      Xbox.setLedOn(LED4);
      Serial.println(F("Down"));
    }
    if (Xbox.getButtonClick(LEFT)) {
      Xbox.setLedOn(LED3);
      Serial.println(F("Left"));
    }
    if (Xbox.getButtonClick(RIGHT)) {
      Xbox.setLedOn(LED2);
      Serial.println(F("Right"));
    }

    if (Xbox.getButtonClick(START)) {
      Xbox.setLedMode(ALTERNATING);
      Serial.println(F("Start"));
    }
    if (Xbox.getButtonClick(BACK)) {
      Xbox.setLedBlink(ALL);
      Serial.println(F("Back"));
    }
    if (Xbox.getButtonClick(L3))
      Serial.println(F("L3"));
    if (Xbox.getButtonClick(R3))
      Serial.println(F("R3"));

    if (Xbox.getButtonClick(LB))
      Serial.println(F("LB"));
    if (Xbox.getButtonClick(RB))
      Serial.println(F("RB"));
    if (Xbox.getButtonClick(XBOX)) {
      Xbox.setLedMode(ROTATING);
      Serial.println(F("Xbox"));
    }

    if (Xbox.getButtonClick(A))
      Serial.println(F("A"));
    if (Xbox.getButtonClick(B))
      Serial.println(F("B"));
    if (Xbox.getButtonClick(X))
      Serial.println(F("X"));
    if (Xbox.getButtonClick(Y))
      Serial.println(F("Y"));
  }
  delay(1);
}
```

Users will need to turn on and connect the controller to the USB-C host shield with a USB cable, before running the example.



8BitDo controller connected to the USB-C Host Shield with a USB-C cable.

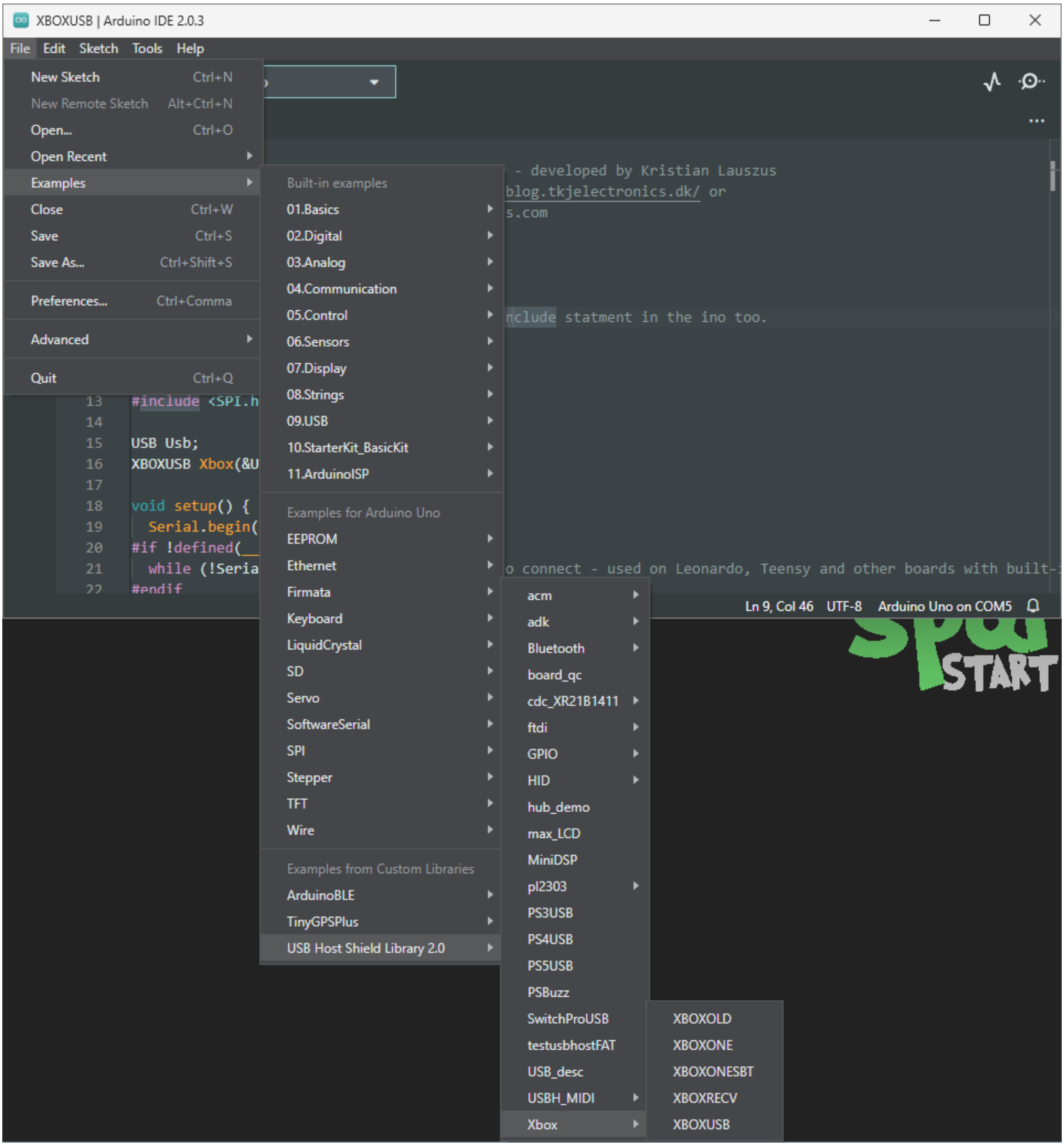
Note

To turn on the controller, press the `Start` + `X` buttons. Users should see two status LEDs blinking at the bottom of the controller.

After the example begins, users should see an output in the Serial Monitor with print out based on the user's interaction with their controller.

Bluetooth Connection

In this example, we will be utilizing the XBOXONESBT example from the USB_Host_Shield_2.0 Arduino library. This example can be found in the **File** dropdown menu *(i.e. (1)* ***File*** *>* ***Examples*** *>* ***USB Host Shield Library 2.0*** *>* ***Xbox*** *>* ***XBOXONESBT****)*. Once the example has been opened, users should see the `XBOXONESBT.ino` example sketch.

1. 

```
 - developed by Kristian Lauszus
blog.tkjelectronics.dk/ or
s.com

nclude statment in the ino too.

#include <SPI.h

USB Usb;
XBOXUSB Xbox(&U

void setup() {
  Serial.begin(
#if !defined(__
  while (!Seria
#endif
o connect - used on Leonardo, Teensy and other boards with built-
```

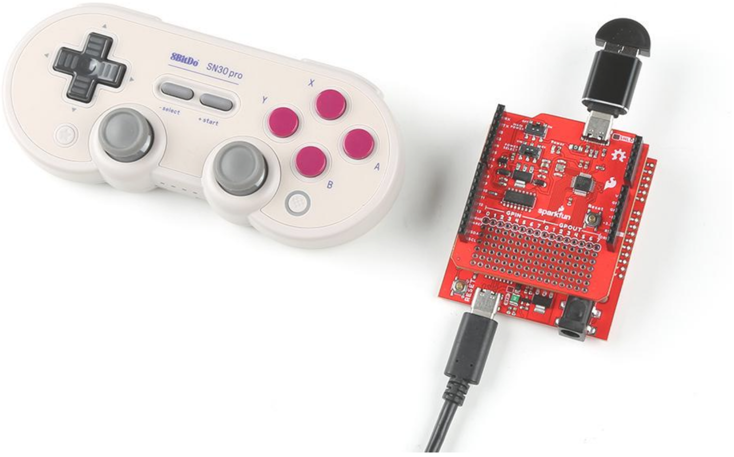
Select the `XBOXONESBT` *example sketch from the* `File` *drop-down menu.*

OXONESBT.ino

```
/*
 Example sketch for the Xbox One S Bluetooth library - developed by Kristian Sloth Lauszus
 For more information visit the Github repository: github.com/felis/USB_Host_Shield_2.0 or
 send me an e-mail:  lauszus@gmail.com
 */

#include <XBOXONESBT.h>
#include <usbhub.h>

// Satisfy the IDE, which needs to see the include statement in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB Usb;
//USBHub Hub1(&Usb); // Some dongles have a hub inside
BTD Btd(&Usb); // You have to create the Bluetooth Dongle instance like so

/* You can create the instance of the XBOXONESBT class in two ways */
// This will start an inquiry and then pair with the Xbox One S controller - you only have to do this once
// You will need to hold down the Sync and Xbox button at the same time, the Xbox One S controller will then start to blink rapidly indicating that it is in pairing mode
XBOXONESBT Xbox(&Btd, PAIR);

// After that you can simply create the instance like so and then press the Xbox button on the device
//XBOXONESBT Xbox(&Btd);

void setup() {
  Serial.begin(115200);
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection
#endif
  if (Usb.Init() == -1) {
    Serial.print(F("\r\nOSC did not start"));
    while (1); //halt
  }
  Serial.print(F("\r\nXbox One S Bluetooth Library Started"));
}
void loop() {
  Usb.Task();

  if (Xbox.connected()) {
    if (Xbox.getAnalogHat(LeftHatX) > 7500 || Xbox.getAnalogHat(LeftHatX) < -7500 || Xbox.getAnalogHat(LeftHatY) > 7500 || Xbox.getAnalogHat(LeftHatY) < -7500 || Xbox.getAnalogHat(RightHatX) > 7500 || Xbox.getAnalogHat(RightHatX) < -7500 || Xbox.getAnalogHat(RightHatY) > 7500 || Xbox.getAnalogHat(RightHatY) < -7500) {
      if (Xbox.getAnalogHat(LeftHatX) > 7500 || Xbox.getAnalogHat(LeftHatX) < -7500) {
        Serial.print(F("LeftHatX: "));
        Serial.print(Xbox.getAnalogHat(LeftHatX));
        Serial.print("\t");
      }
      if (Xbox.getAnalogHat(LeftHatY) > 7500 || Xbox.getAnalogHat(LeftHatY) < -7500) {
        Serial.print(F("LeftHatY: "));
        Serial.print(Xbox.getAnalogHat(LeftHatY));
        Serial.print("\t");
      }
      if (Xbox.getAnalogHat(RightHatX) > 7500 || Xbox.getAnalogHat(RightHatX) < -7500) {
        Serial.print(F("RightHatX: "));
        Serial.print(Xbox.getAnalogHat(RightHatX));
        Serial.print("\t");
      }
      if (Xbox.getAnalogHat(RightHatY) > 7500 || Xbox.getAnalogHat(RightHatY) < -7500) {
        Serial.print(F("RightHatY: "));
        Serial.print(Xbox.getAnalogHat(RightHatY));
      }
      Serial.println();
    }

    if (Xbox.getButtonPress(LT) > 0 || Xbox.getButtonPress(RT) > 0) {
      if (Xbox.getButtonPress(LT) > 0) {
        Serial.print(F("LT: "));
        Serial.print(Xbox.getButtonPress(LT));
        Serial.print("\t");
      }
      if (Xbox.getButtonPress(RT) > 0) {
        Serial.print(F("RT: "));
        Serial.print(Xbox.getButtonPress(RT));
        Serial.print("\t");
      }
      Serial.println();
    }

    // Set rumble effect
    static uint16_t oldLTValue, oldRTValue;
    if (Xbox.getButtonPress(LT) != oldLTValue || Xbox.getButtonPress(RT) != oldRTValue) {
      oldLTValue = Xbox.getButtonPress(LT);
      oldRTValue = Xbox.getButtonPress(RT);
      uint8_t leftRumble = map(oldLTValue, 0, 1023, 0, 255); // Map the trigger values into a byte
      uint8_t rightRumble = map(oldRTValue, 0, 1023, 0, 255);
      if (leftRumble > 0 || rightRumble > 0)
        Xbox.setRumbleOn(leftRumble, rightRumble, leftRumble, rightRumble);
      else
        Xbox.setRumbleOff();
    }

    if (Xbox.getButtonClick(UP))
      Serial.println(F("Up"));
    if (Xbox.getButtonClick(DOWN))
      Serial.println(F("Down"));
    if (Xbox.getButtonClick(LEFT))
```

```
      Serial.println(F("Left"));
    if (Xbox.getButtonClick(RIGHT))
      Serial.println(F("Right"));

    if (Xbox.getButtonClick(VIEW))
      Serial.println(F("View"));
    if (Xbox.getButtonClick(MENU))
      Serial.println(F("Menu"));
    if (Xbox.getButtonClick(XBOX)) {
      Serial.println(F("Xbox"));
      Xbox.disconnect();
    }

    if (Xbox.getButtonClick(LB))
      Serial.println(F("LB"));
    if (Xbox.getButtonClick(RB))
      Serial.println(F("RB"));
    if (Xbox.getButtonClick(LT))
      Serial.println(F("LT"));
    if (Xbox.getButtonClick(RT))
      Serial.println(F("RT"));
    if (Xbox.getButtonClick(L3))
      Serial.println(F("L3"));
    if (Xbox.getButtonClick(R3))
      Serial.println(F("R3"));

    if (Xbox.getButtonClick(A))
      Serial.println(F("A"));
    if (Xbox.getButtonClick(B))
      Serial.println(F("B"));
    if (Xbox.getButtonClick(X))
      Serial.println(F("X"));
    if (Xbox.getButtonClick(Y))
      Serial.println(F("Y"));
  }
}
```

Users will need to connect the Bluetooth USB module to the USB-C host shield with the USB adapter before running the example. After the example begins, users should see an output in the Serial Monitor with print out based on the user's interaction with their controller.



Bluetooth module connected to the USB-C Host Shield; and paired with an 8BitDo controller.

!! note Make sure to wait until after the board restarts and executes the example, before pairing the 8BitDo controller with the Bluetooth module.

Bluetooth Pairing the Controller

To turn on the controller, press the `Start` + `X` buttons. Users should see two status LEDs blinking at the bottom of the controller. To pair the controller, press and hold the pair button at the top of the controller, next to the USB-C connector, for 3 seconds. Once paired, the controller should vibrate.

2023-03-04

2023-03-04

santaimpersonator

GitHub

2. Resources

2.1 Product Resources

- Product Page
- Schematic (PDF)
- Eagle Files (ZIP)
- Board Dimensions (PDF)
- Arduino Library: USB Host Rev. 2.0
- GitHub Hardware Repo

2.1.1 Additional Resources

- Arduino Shields Tutorial (v2)
- Arduino Shields Product Category
- SparkFun Technical Assistance

2.2 Hardware Component Documentation

- USB Peripheral/Host Controller: MAX3421E (PDF)
- Errata_MAX3421E (PDF)
- Programming Guide (PDF)
- Technical Articles
- Article - Turn any video game controller into a USB mouse (PDF)
- Application Notes
- The Maxim USB Laboratory (PDF)
- Setting Up the Maxim USB Laboratory (PDF)
- Power Regulation:
- MIC5205 (PDF)
- LM1117 (PDF)
- Logic-Level Converter:
- 74HC4050 (PDF)

2.3 Manufacturer's Resources

Maxim Integrated *(now part of Analog Devices)* also provides great resources for the MAX3421E USB Peripheral/Host Controller:

- MAX3421E Product Page
- Technical Documentation
- Tutorial - Turn any video game controller into a USB mouse
- Technical Support Page
- Knowledge Base Page

2023-03-04

2023-03-04

santaimpersonator

GitHub

3. Support

3.1 Troubleshooting Tips

Need Help?

If you need technical assistance or more information on a product that is not working as you expected, we recommend heading on over to the SparkFun Technical Assistance page for some initial troubleshooting.

SparkFun Technical Assistance Page

If you can't find what you need there, the SparkFun Forums is a great place to search for additional information and to ask questions.

Account Registration Required

If this is your first visit to our forum, you'll need to register an account to post questions.

3.1.1 Initialization Failure

The following error message, in the serial terminal, indicates that there was a problem communicating with the MAX3421E chip.

```
OSC did not start
```

This error occurs here in the example code:

```
if (Usb.Init() == -1)
    Serial.println("OSC did not start.");
```

Here are a few steps users can perform to diagnose the issue:

- Double-check the hardware connections; including, but not limited to the solder joints, header pins (male and female), etc.
- Disconnect power from the board and try a continuity test with a multimeter.
- Make sure the switches are in the correct position to provide power to the shield.
- The shield requires a minimum 5V input voltage.
- The red, power LED should be lit when the shield is powered.
- Double-check the library for any I/O pin modifications.

3.1.2 USB Hub

If users connect USB hubs or USB cables with a hub to the USB host shield, refer to the hub_demo example from the USB_Host_Shield_2.0 Arduino library. This example can be found in the **File** dropdown menu *(i.e. **File > Examples > USB Host Shield Library 2.0 > hub_demo**)* and will list the USB description for the hub(s) and all the peripheral devices connected to the hub(s).

Only interested in the USB hub description?

To see just the USB description for the hub(s) connected to the USB host shield, follow the information in the library's FAQ. Utilizing the USB_dec example, uncomment lines 12-18(1).

1. Each instance of `USBHub Hub<number>(&Usb);` enables a USB hub, but the library is limited up to **seven** USB hubs.

```
11  USB     Usb;
12  //USBHub  Hub1(&Usb);
13  //USBHub  Hub2(&Usb);
14  //USBHub  Hub3(&Usb);
15  //USBHub  Hub4(&Usb);
16  //USBHub  Hub5(&Usb);
17  //USBHub  Hub6(&Usb);
18  //USBHub  Hub7(&Usb);
```

2023-03-04

2023-03-04

santaimpersonator

GitHub